


[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide

 SEARCH

THE ACM DIGITAL LIBRARY


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

Fast breakpoints: design and implementation

Full text Pdf (855 KB)

Source [Conference on Programming Language Design and Implementation](#) [archive](#)
Proceedings of the ACM SIGPLAN 1990 conference on Programming language design and implementation [table of contents](#)
 White Plains, New York, United States
 Pages: 78 - 84
 Year of Publication: 1990
 ISSN:0362-1340
[Also published in ...](#)

Author [Peter B. Kessler](#)
Sponsor [SIGPLAN](#): ACM Special Interest Group on Programming Languages

Publisher ACM Press New York, NY, USA

Additional Information: [abstract](#) [references](#) [citing](#) [index terms](#) [collaborative colleagues](#)

Tools and Actions: [Discussions](#) [Find similar Articles](#) [Review this Article](#)
[Save this Article to a Binder](#) [Display in BibTex Format](#)

DOI Bookmark: Use this link to bookmark this Article: <http://doi.acm.org/10.1145/93542.93555>
[What is a DOI?](#)

↑ ABSTRACT

We have designed and implemented a fast breakpoint facility. Breakpoints are usually thought of as a feature of an interactive debugger, in which case the breakpoints need not be particularly fast. In our environment breakpoints are often used for non-interactive information gathering; for example, procedure call count and statement execution count profiling [Swinehart, et al.]. When used non-interactively, breakpoints should be as fast as possible, so as to perturb the execution of the program as little as possible. Even in interactive debuggers, a conditional breakpoint facility would benefit from breakpoints that could transfer to the evaluation of the condition rapidly, and continue expeditiously if the condition were not satisfied. Such conditional breakpoints could be used to check assertions, etc. Program advising could also make use of fast breakpoints [Teitelman]. Examples of advising include tracing, timing, and even animation, all of which should be part of an advanced programming environment. We have ported the Cedar environment from a machine with microcode support for breakpoints [Lampson and Pier] to commercial platforms running C code [Atkinson, et al.]. Most of our ports run under the Unix* operating system, so one choice for implementing breakpoints for Cedar was to use the breakpoint facility provided by that system. The breakpoints provided by the Unix operating system are several orders of magnitude too slow (and also several process switches too complicated) for the applications we have in mind. So we designed a breakpoint system that was fast enough for our purposes. Breakpoints for uni-processors running single threads of control used to be fast and simple to implement. This paper shows that breakpoints can still be fast, even with multiple threads of control on multi-processors. This paper describes problems in the design of a breakpoint package for modern computer architectures and programming styles, and our solutions to them for a particular architecture.

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

Aral and Gertner Ziya Aral , Ilya Gertner, High-level debugging in parasight, ACM SIGPLAN Notices, v.24 n.1, p.151-162, Jan. 1989

Atkinson, et al. R. Atkinson , A. Demers , C. Hauser , C. Jacobi , P. Kessler , M. Weiser, Experiences creating a portable cedar, ACM SIGPLAN Notices, v.24 n.7, p.322-329, July 1989

Demers, et al. Alan Demmers , Mark Weiser , Barry Hayes , Hans Boehm , Daniel Bobrow , Scott Shenker, Combining generational and conservative garbage collection: framework and implementations, Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.261-269, December 1989, San Francisco, California, United States

Evans and Darley T. G. Evans and D. L. Darley, "On-line Debugging Techniques: A Survey", in Proceedings of the Fall Joint Computer Conference, 1966.

Fabry R. S. Fabry, "MADBUG - A MAD Debugging System", in The Compatible Time-Sharing System. A Programmer's Guide, Second Edition, MIT Press, Cambridge MA, 1965.

Gill S. Gill, "The Diagnosis of Mistakes in Programmes on the EDSAC", in Proceedings of the Royal Society, Series A, Vol. 206, pp. 538-554.

Graham, et al. Susan L. Graham , Peter B. Kessler , Marshall K. Mckusick, Gprof: A call graph execution profiler, Proceedings of the 1982 SIGPLAN symposium on Compiler construction, p.120-126, June 23-25, 1982, Boston, Massachusetts, United States

Johnson M. S. Johnson, An Annotated Software Debugging Bibliography, Hewlett-Packard Laboratories, March 1982.

Kane Gerry Kane, MIPS RISC architecture, Prentice-Hall, Inc., Upper Saddle River, NJ, 1988

Klensk and Larson T. J. Klensk and L. E. Larson, IBM Technical Disclosure Bulletin, Vol. 15, No. 4, pp 1248-50, September 1972.

Knuth Donald E. Knuth, The art of computer programming, volume 1 (3rd ed.): fundamental algorithms, Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, 1997

Lampson and Pier Butler W. Lampson , Kenneth A. Pier, A processor for a high-performance personal computer, Proceedings of the 7th annual symposium on Computer Architecture, p.146-160, May 06-08, 1980, La Baule, United States

Pappas and Murray Chris H. Pappas , William H. Murray, III, Eighty-Three Eighty-Six Microprocessor Handbook, Osborne/McGraw-Hill, Berkeley, CA, 1987

Stockham and Dennis T. G. Stockham and J. B. Dennis, "FLIT- Flex0writer Interrogation Tape: A Symbolic Utility Program for the TX-0", Memo 5001-23, Department of Electrical Engineering, MIT, July 1960.

Sun Debugging Sun Microsystems, Inc., Debugging Tools, Part No. 800-1775-10, May 1988.

Sun SPARC Sun Microsystems, Inc., The SPARC Architecture Manual, Part No. 800-1399-03, June 1987.

Swinehart, et al. Daniel C. Swinehart , Polle T. Zellweger , Richard J. Beach , Robert B. Hagmann, A structural view of the Cedar programming environment, ACM Transactions on Programming Languages and Systems (TOPLAS), v.8 n.4, p.419-490, Oct. 1986

Teitelman W. Teitelman, Interlisp Reference Manual, Xerox Corporation, Palo Alto, CA, Oct, 1978.

↑ CITINGS 17

Bob Boothe, Efficient algorithms for bidirectional debugging, ACM SIGPLAN Notices, v.35 n.5, p.299-310, May 2000

Thomas Ball, Efficiently counting program events with support for on-line queries, ACM Transactions on Programming Languages and Systems (TOPLAS), v.16 n.5, p.1399-1410, Sept. 1994

Norman Ramsey, Correctness of trap-based breakpoint implementations, Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.15-24, January 16-19, 1994, Portland, Oregon, United States

David Keppel, A portable interface for on-the-fly instruction space modification, ACM SIGARCH Computer Architecture News, v.19 n.2, p.86-95, Apr. 1991

Bartosz Baliś , Marian Bubak , Włodzimierz Funika , Roland Wismüller, A monitoring system for multithreaded applications, Future Generation Computer Systems, v.19 n.5, p.641-650, July 2003

A. Singhal , A. J. Goldberg, Architectural support for performance tuning: a case study on the SPARCcenter 2000, ACM SIGARCH Computer Architecture News, v.22 n.2, p.48-59, April 1994

Daniel Schulz , Frank Mueller, A thread-aware debugger with an open interface, ACM SIGSOFT Software Engineering Notes, v.25 n.5, p.201-211, Sept. 2000

Ali-Reza Adl-Tabatabai , Thomas Gross, Detection and recovery of endangered variables caused by instruction scheduling, ACM SIGPLAN Notices, v.28 n.6, p.13-25, June 1993

Robert Wahbe, Efficient data breakpoints, ACM SIGPLAN Notices, v.27 n.9, p.200-212, Sept. 1992

Peter A. Buhr , Martin Karsten , Jun Shih, KDB: a multi-threaded debugger for multi-threaded applications, Proceedings of the SIGMETRICS symposium on Parallel and distributed tools, p.80-87, May 22-23, 1996, Philadelphia, Pennsylvania, United States

K. Scott , N. Kumar , S. Velusamy , B. Childers , J. W. Davidson , M. L. Soffa, Retargetable and reconfigurable software dynamic translation, Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization, p.36, March 23-26, 2003, San Francisco, California

Robert Wahbe , Steven Lucco , Susan L. Graham, Practical data breakpoints: design and implementation, ACM SIGPLAN Notices, v.28 n.6, p.1-12, June 1993

Norman Ramsey , David R. Hanson, A retargetable debugger, ACM SIGPLAN Notices, v.27 n.7, p.22-31, July 1992

Mustafa M. Tikir , Jeffrey K. Hollingsworth , Guei-Yuan Lueh, Recompilation for debugging support in a JIT-compiler, ACM SIGSOFT Software Engineering Notes, v.28 n.1, January 2003

Jaydeep Marathe , Frank Mueller , Tushar Mohan , Bronis R. de Supinski , Sally A. McKee , Andy Yoo, METRIC: tracking down inefficiencies in the memory hierarchy via binary rewriting, Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization, p.289, March 23-26, 2003, San Francisco, California

Clara Jaramillo , Rajiv Gupta , Mary Lou Soffa, Comparison checking: an approach to avoid debugging of optimized code, ACM SIGSOFT Software Engineering Notes, v.24 n.6, p.268-284, Nov. 1999

Peter M. Chen , Wee Teck Ng , Subhachandra Chandra , Christopher Aycock , Gurushankar Rajamani , David Lowell, The Rio file cache: surviving operating system crashes, ACM SIGPLAN Notices, v.31 n.9, p.74-83, Sept. 1996

↑ INDEX TERMS

Primary Classification:

D. Software

↳ D.2 SOFTWARE ENGINEERING

↳ D.2.5 Testing and Debugging

↳ **Subjects:** Debugging aids

Additional Classification:

D. Software

↳ D.2 SOFTWARE ENGINEERING

↳ D.3 PROGRAMMING LANGUAGES

↳ D.3.2 Language Classifications

↳ **Nouns:** Cedar

↳ D.3.3 Language Constructs and Features

↳ **Subjects:** Procedures, functions, and subroutines

General Terms:

Design, Languages, Performance, Reliability

↑ Collaborative Colleagues:

Peter B. Kessler: Susan L. Graham

Marshall K. Mckusick

↑ This Article has also been published in:

- ACM SIGPLAN Notices

Volume 25 , Issue 6 (June 1990)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)